DST analysis for BRAHMS[*]

Ionut-Cristian Arsene

February 15, 2005

Contents

1 Searching for the data files you need (LTRs and DSTs)

The first thing to do when you want to make an analysis, you must search for some data. So, you must find the proper runs for your job. For this, you can use the scripts from
http://brahms-web.brahms.bnl.gov/daq/.

[*]90% of what it's written here can be found in the README files maked by Djamel Ouerdane, Peter Christiansen, Claus Jorgensen and Pawel Staszel. So, what I did is just puting together some useful information.

You can find there all the needed information (run settings, number of events, files location, etc.). After you located the files, it is recommended that you make some symbolic links pointing at them. This can be done with

```
bash> ln -s /path/to/the/file link_name
```

If you need to make a big number of symbolic links, you can use some regular expressions with the *ln* command.

2 Efficiency

The efficiency calculations are made with Pawel Staszel's package (only for the FS), which can be obtained from the CVS with the command:

```
bash> cvs co brahms_app/ps_app
```

Make sure you have the *CVSROOT* set correctly. The MRS tracking efficiency is computed using Truls's curves.

Pawel's package allows one to find the track reconstruction efficiency for each of the BRAHMS forward arm tracking detector (T1, T2, T3, T4, T5), and for the time of flight detectors H1 and H2. For more details you can read the README files from the package.

One important thing, before start launching the efficiency jobs is: look if someone else already did this for your specific settings, because the efficiency jobs can be very time consuming. You can just copy that file into your analysis directory.

If nobody has done yet the efficiencies that you need, you must generate them yourself. This is done in several steps. The first thing that you need is the LTR directory for your settings (the LTR files serve as an input for the efficiency package). So, let's start:

Step 1 Create the matching offsets for detector combinations. This offsets are different than gtr offsets and are required for the proper efficiency determination. To do this, type:

```
> bratmain fsRecEff*.C -r 11301 -n 20 --prod=1 --ver=0
        --offsetmode=1 --iroot=10 --period=run04
        --spec=auau --energy=63
```

In the above command we just launched an efficiency script for run 11301, using maximum 20 LTR sequences with production number 1 and version 0. The parameters *–offsetmode* and *–iroot* must be 1 and 10 respectively for the offset job. The offsets and sigmas will be stored into a file called *EfficParamFile_NEW.root*. If you have problems in finding this file, look in the directory *brahms_app/ps_app/fs/receffnew/detmatch*. A file with diagnostic histograms called *run11301-10.root* is also produced. To make sure the parameters are right is to make a look at the diagnostic histograms in the above file. If you are sure that the parameters are right you have to copy them from *EfficParamFile_NEW.root* into *EfficParamFile.root*. Use the script *detmatch/UpdateEfficParamFile.C* for this purpose:

```
    > root
root> .L UpdateEfficParamFile.C
root> AddRun(11301)
```

Note that you can add more runs at once. See the list of formal arguments.

Step 2 The production run. The command line for the production job should look like bellow

```
> bratmain fsRecEff*.C -r 11301 -n 20 --prod=1 --ver=0
        --offsetmode=0 --iroot=1 --period=run04
        --spec=auau --energy=63 --tree=true
```

This should produce an output tree in file *root11301-1.root*.

Step 3 Analisys of the tree and storage of the efficiency results in the form of 3D and 1D histograms. o to *scripts/* directory and do:

```
    > root
root> .L ../BrRefTrack.h
root> .L MapEfficiency.C
root> MapEfficiency("run04",63,8,861,"T1",1,"CXAX",4)
```

This will show the efficiency for T1 detector versus centrality, x detector position and ax (slope parameter). Use the tab completion to see the entire list off arguments. If you want to store the final results (3D and 1D histograms) do:

```
root> MapEfficiency("run04", 63, 8, 861, "T1", 1, "CXAX",
                     4, 1000000, kTRUE, kTRUE)
```

The above command must be repeated for T2, T3, T4, T5, H1, and H2 for every setting that you want to take into account. The histograms will be stored in a file named *MappedEffic_run04_auau_63.root*. Note that the above commands are just examples. The arguments are not necessarily correct.

Step 4 Files like *MappedEffic_run04_auau_63.root* should be used in the analysis programs for reconstruction efficiency corrections.

A last suggestion about the efficiency calculations is connected with the statistics. You need a reasonable number of input events. So, I think you need at least two big runs per setting or more smaller runs. In this way you will have more than one output file (see Step 2) per setting. This is not a problem, because the scripts from step 3 are looking in all the files corresponding to that setting.
If you still have problems about this subject, you can contact Pawel Staszel at ufstasze@if.uj.edu.pl

3 Acceptance maps

The acceptance package contains spectrometer 'geometry' classes (FS and MRS) and an acceptance map class. The geometry classes contain methods for swimming lines (BrLine3D) through detectors and magnets, checks where these lines intersect the hodoscopes and return a status (boolean, slat number) that depends on whether the line is within the acceptance (fiducial cuts, slat range).

The binary that make the acceptance maps is called *generateMaps* and can be found in *brahms_app/analyze_app/*. Everything is self explanatory if you type:

```
    > generateMaps -h
Usage: generateMaps [options]

Options:
 -h  --help             Show this help, default is true
 -e  --maxError         Maximum average error on pion map,
                               default is 5
 -o  --outputfile       Output file name, default is
 -r  --run              Run number, default is 5361
 -s  --spectrometer     Spectrometer (FFS,FS,MRS), default is
 -V  --version          Version number, default is false
 -d  --vtxbinwidth       Width of vertex bins, default is 5
 -Z  --vtxzmax           Max. vertex z, default is 20
 -z  --vtxzmin           Min. vertex z, default is -20
```

So, a small example of a job that generates an acceptance map is the following:

```
   > generateMaps -e 4 -o accMap8B861.root -r 11301 -s FS
```

This will generate an acceptance map for setting 8B861 (FS at 8 degrees, D1 current at 861 and polarity B). Note that the above mentioned settings are passed to the executable trough the run number 11301, which has those settings. The executable will interrogate the BRAHMS run database to find the settings for the specified run.

One must generate one acceptance map for every setting analysed.

4 Making microDSTs

The microDSTs are generated for each spectrometer setting and contain information for a spectrometer analysis. This is motivated by the fact that DSTs contain too much info and complex data types, which make them slow and heavy to use. By having microDSTs with just the info you need, stored in simple data types (int, float), analyzing the data becomes less frustrating and more efficient.

The programs neccessary to make microDSTs are *dst2mrstree* (for MRS) and *dst2fstree* (for FS). Type

```
   > dst2mrstree -h
```

and

```
   > dst2fstree -h
```

for help.

The microDSTs are generated in two steps, meaning that there is an aditional preprocess step. This step is for evaluating some offsets and distributions widths for later data cuts. The preprocess is done by using the *-p* option, while for truly generating the microDSTs, you need to use the same command line without *-p* option. So, the command

```
  > dst2mrstree -p -S 40A1050 -S 45A1050 -S 90B350
           -o mrspre -s mrspar -i ~/data
```

means that you want to preprocess (-p) the DSTs for the MRS settings (-S) 40A1050, 45A1050 and 90B350. The output histograms will be saved in files like (-o) *mrspre<setting>.root* and the fit results and parameters will be saved in files (-s) like *mrspar<setting>.sel*. The input files (-i) are located in the directory */data*.

The *dst2fstree* program is used in a similar way.

5 Making data maps (a la NBI)

In this section, we describe a method of analysis maked by Peter Christiansen and Djamel Ouerdane at NBI. This method[1] is suitable for soft physics analysis (identified particle yields).

This method consists of 4 steps:

Step 0 This step is necessary only for MRS. We must 'split' the MRS acceptance maps, because the analysis software maked by PC and DO doesn't handle the acceptance maps maked 'a la Claus'[2]. So, we must transform the acceptance maps generated with *generateMaps* into new acceptance maps. To do this, type:

```
    > splitMrsmap -i <old map> -o <new map>
```

Step 1 Generate 2D data and correction histograms (y, p_T) for $\pi^{+/-}$, $K^{+/-}$ and $p^{+/-}$ with the same binning as acceptance maps. For this you have to use the *mrstree2datamap* and *fstree2datamap* programs. The *-h* option implemented for both programs gives you useful information:

```
    > mrstree2datamap -h
Usage: mrstree2datamap [options]

Options:
 -a  --acceptance-map-basename  Acceptance map basename,
                                 default is accmap
 -C  --cent-max      Centrality high cut, default is 20
 -c  --cent-min      Centrality low cut, default is 0
 -e  --event         Number of events, default is 100000000
 -h  --help          Show this help, default is true
```

[1]The scripts neccessary for this analysis can be found in the analyze_app package, in the method1 directory

[2]Claus introduced MRS maps where negative particles have a negative p_T.

```
-b  --input-base    Root input BASENAME filename,
                         default is mrstree
-o  --output-base   Root output BASENAME filename,
                         default is mrsdata
-p  --pid           Detector name (TOFW), default is TOFW
-S  --spectrometer-setting   Spectrometer setting
                             (e.g. 45B700), default is
-V  --version       Version number, default is false
```

For example, the MRS data can be processed this way:

```
> mrstree2datamap -b mrstree -S 90A350 -S 90A1050 -p TOFW
             -c 0 -C 20 -a mrsmap -o mrsdata
```

which means: generate MRS data maps of basename (-o) *mrsdata* using microDSTs of basename (-b) *mrstree* for MRS settings (-S) *90A350* and *90A1050*, using PID device (-p) *TOFW* between centrality (-c) 0 and (-C) 20, using acceptance map basename (-a) *mrsmap*.

The output files will be *mrsdata90A350.root* and *mrsdata90A1050.root*. They contain diagnostic histograms and data maps (raw yields + efficiency/correction maps) that are designed the same way acceptance histograms are.

Note: In the FS we have more PID devices, so make sure that you provide meaningful names for your output files[3].

Step 2 Now it is the time to use the macros for building up spectra. First, there is a necessary setup for using them in a ROOT session. I suggest to write a small script that must contain the following:

```
{
  TString libPath("~/brahms_app/analyze_app/lib/banana");
  TString macroPath("~/brahms_app/analyze_app/share/banana/scripts");

  gSystem->Load(Form("%s/libSelection.so",libPath.Data()));
  gSystem->Load(Form("%s/libTreeReader.so",libPath.Data()));
  gSystem->Load(Form("%s/libAcceptance.so",libPath.Data()));
  gSystem->Load(Form("%s/libSpectraObject.so",libPath.Data()));
```

[3]The output file name must contain the name of the PID device (e.g. h1data<setting>.root)

```
  gSystem->Load(Form("%s/mrsSpectra.C",macroPath.Data()));
  gSystem->Load(Form("%s/fsSpectra.C",macroPath.Data()));
  gSystem->Load(Form("%s/plotResultsHelp.C",macroPath.Data()));
  gSystem->Load(Form("%s/plotResults.C",macroPath.Data()));
}
```

When you start a root session, the first you must do is to load this script.

Ok, now the idea is to generate a 'SpectraObject'. Remember, we have created some data maps called *mrsdata<setting>.root*. The macro *mrsSpectra.C* will create an object containing all necessary histograms (data, acceptance and corrections). This object has methods that are used for applying the correction to the data, make the proper weighted average (vertex binning, spectrometer settings) and projection to p_T axis. Here is how to use the mrsSpectra.C :

```
  mrsSpectra("<setting>",     // e.g. 90B350
              <pid>,          // (Int_t) +/-1 = pi+/-,
                              //         +/-2 = K +/-,
                              //         +/-3 = p/pbar
             "<input directory>", // default is "."
              <dead slat>,    // (Bool_t) use or not acc
                              //    maps with dead slats
              <accCut>,       // (Float_t) minimum acc
                           //    value (remove map edges)
              <pmin>,      // minimum momentum (not pT!!!)
              <pmax>)      // maximum momentum (not pT)
```

Example:

```
root> mrsSpectra("90B350", -2, ".", kFALSE, 0.005, 0.2, 2.0)
```

will get you a file called *akaon90B350.root* containing a SpectraObject object with K- data for the setting 90B350. The FS stuff is similar, except that you need to specify which PID detector you used.

Note: Be careful in the FS. If you want to combine RICH, H2 and H1 data later on (step 3), be sure that momentum ranges do not overlap when the data come from the same FS setting. If you are not careful enough, statistical errors will be wrong in the overlap of the momentum ranges since you will count the particles more than once when averaging between the different momentum sets.

Step 3 Now that you have all SpectraObject root files for a given centrality class, it is time to overlay them, averaging the overlap zones (different spectrometer settings), slicing the rapidity axis, projecting to the *pT* axis and fitting your spectrum to get yields and slopes.

First thing is to load the different data sets you have created before for a given particle:

```
root> addSetting("<data directory>", "<setting>",
                 "<detector>", <pid>)
```

FS example:

```
root> addSetting(".","all","rich",1) // for pi+ from RICH
root> addSetting(".","all","h2",1)   // for pi+ from H2
root> addSetting(".","all","h1",1)   // for pi+ from H1
```

MRS example:

```
root> addSetting(".", "all", "", 1)     // for pi+
```

The 'all' option take all spectrometer settings that are available. In the MRS, note that the detector name is an empty string (for now, only TOFW is implemented).

You can list all data sets:

```
root> listSettings()
```

You can remove some particular settings:

```
root> rmSettings("<setting>")
```

Now comes the fun part: overlaying stuff, slicing, projecting and fitting. This is all done in a single function call:

```
plotResults("<data dir>",   // (const char*)input directory
            "<outputname>", // (const char*)output basename
             <minrap>,      // (float) minimum rapidity
             <maxrap>,      // (float) maximum rapidity
             <minpT>,       // (float) minimum pT
             <maxpT>,       // (float) maximum pT
             <fitOpt>,      // (int) fit function
             <rebin>)       // (int) rebinning of histogram
```

Example:

```
root> plotResults(".", "testpiplusy0", -0.1, 0.1,
                  0.2, 2.0, 5, 2)
```

The $< fitOpt >$ it a code for the function that you want to use for fitting the spectra. $<$ *rebin* $>$ is a code used for the *Rebin* method of the histograms.

6 For the end...

At this point, if everything goes well, you should have the yields for a specific particle, in a specific rapidity range. The analysis package contains programs and scripts only up to this point (but I think it's more than enough). If you want to do more or something else, you can start making your own scripts :).

So, here at the end of this document, I want to point out that these programs are made by few people for their own purposes and I recommend to read them carefully before use them. You can end up with some errors at anytime and if you do not know what you are running it's going to be hard to repair the problems. Anyway, this piece of analysis software, as it is, it's a great help for everyone.

7 Appendix

7.1 Getting data from HPSS

HPSS is the place where our large files (e.g. raw data) are stored. You can take that data from HPSS and transfer it to your directory using commands very similar to the usual bash. So, to start you need to type

```
bash> hsi
```

in your prompt. After authentication, you will obtain a new prompt. The current directory will be /, so you need to move trough directories using *cd* . Our raw data is located in the directory */home/bramsink/raw/*. To get to a specific run, starting from the *raw* directory you need to

```
> cd 2005/02/17/
```

This will take you to the directory containing the runs acquired on 17 february 2005. But you can use the *ls* command to guide yourself. To download files from there, you can use the *get* command.

7.2 Contact me

This document it's not complete because the problems are many and the time is limited, so if you have troubles and tryied for hours to solve them without succes :)), contact me at the following mailing addresses:

aic@venus.nipne.ro

ionut.arsene@gmail.com

ionutcristianarsene@yahoo.com.

Let's hope that our minds putted together will overcome any obstacle standing between you and the true scientific knowledge :D.